The Parameters Tuning for Evolutionary Synthesis Algorithm

Gregor Papa and Jurij Šilc
Computer Systems Department
Jožef Stefan Institute
Jamova c. 39
SI-1000 Ljubljana
Slovenia
gregor.papa@ijs.si, jurij.silc@ijs.si, http://csd.ijs.si

Keywords: evolutionary, scheduling, allocation, genetic operators, tuning

Received: January 29, 2003

This paper covers the evaluation and fine-tuning of different values of genetic operator's parameters in the process of optimizing the designs of the integrated circuits. We investigated the interdependence of various values of these parameters in the use over the set of test-bench circuits, as well as their influence on the quality of the final solution and the convergence speed. Due to the increasing usage of the evolutionary optimization in the area of the integrated circuit design, there is a need to find a proper combination of genetic operators parameters' value to make optimal solutions. Therefore, it is important to perform this kind of evaluation for each new problem to be solved.

1 Introduction

The area of evolutionary computation is very popular but there is always a problem of defining a proper value of parameters of genetic operators. A standard genetic algorithm uses four different parameters that have to be defined in advance, before the algorithm is actually used. These are: the number of generations, the size of the population, the probability of crossover, and the probability of mutation [1].

There are some proposals for setting of these parameters according to the problem size and according to the area of the problem. But these proposals are not always applicable or are not suitable for all problems. Also, there are no proposals for any additional operators, used in some optimizations, which improve the performance of the algorithm.

To find some dependencies between the parameters and the problem that has to be solved, we made the evaluation, similar to that in [7]. We study an evolutionary approach that automatically generates circuit designs. We managed to point to some interesting dependencies between parameters themselves and to determine what values should be used in our optimizations when working with evolutionary-oriented algorithms.

2 ECSA algorithm

The facts presented in the introduction paragraphs and promising results of different evaluations [4, 9, 10] took us to the Evolutionary Concurrent Scheduling and Allocation (ECSA) design approach [8]. This approach considers scheduling and allocation constraints, allows short design time and can find globally optimal solutions. The input description of the integrated circuit (IC) is transformed into two basic (initial) schedules, obtained by As-Soon-As-Possible and As-Late-As-Possible algorithms. Functional units (FUs) used in first case are those fastest for each operation and in second case those slowest for each operation. These two schedules present some kind of boundary solutions, since all other solutions are executed in-between the time limits defined by these two schedules. Namely, no other solution can be faster or slower, considering different combinations of used units.

Each solution has to be properly encoded (into the chromosome), i.e., each operation's start time and FU have to exist in the chromosome. Initial population is built upon the two initial solutions, which are multiplied to form the population with so-called boundary solutions. The optimal solution has to be somewhere in-between the boundaries, therefore genetic operators (crossover, mutation, variation) transform those encoded solutions. With transformations their start times and allocated FUs are changed. The final solution obtained by genetic operators is also influenced by simulated annealing algorithm [6], which improves the solution if it stopped somewhere near the globally optimal point.

2.1 Encoding

The chromosome string consists of the numbers that represent the starting time of each operation and the allocated unit for each operation, where the position in the string depends on the order of the operations in the input IC description. This means that the chromosome consists of pairs of time/space information for each operation. And the genetic operators can influence both parts of that information, either together or separately.

The selected encoding type is chosen because of its convenience. When strings have to be further transformed, checked and analyzed, there is no need for any additional conversion of their values. In addition, the used implementation of genetic operators can check the changed values (their feasibility) instantly, without any transformation. The correctness of the transformation can therefore be checked within the function itself.

2.2 Cost function

One of the most important parts of the algorithm is its cost function. To obtain the cost (Eq. 1) of a certain circuit, the algorithm has to evaluate the required number of resources. In contrast to the other multi-objective functions that give more than one final solution, this one already includes the decision making part, which chooses one solution form all the solutions on the Pareto front.

$$
\begin{aligned}
Cost &= \sqrt{\sum_{i=1}^{N}(cost_{f_i})^2 + cost_r^2 + cost_b^2 + cost_t^2} \\
cost_{f_i} &= w_{f_i} F_i \\
cost_r &= w_r n_r \\
cost_b &= w_b n_b \\
cost_t &= w_t T
\end{aligned}
\qquad (1)
$$

The elements of the function above are calculated as follows.

The number n_{fi} is the highest number of the i-th functional unit needed in a separate control step.

The number n_r is the highest number of variables needed in a separate control step. We consider variables that are needed by the functional unit as input data, variables that are returned as output data, and variables that are not used at the moment but will be used in some of the later control steps or must be available until the end of the execution of all operations.

The number n_b is the highest number of data transmissions (into or from the functional units) in a separate moment.

The execution time, T, is the time needed to execute all the operations of the schedule.

The weights w_{fi}, w_r, w_b, and w_t are the weights of functional units, registers, buses and time, respectively, to be considered in the IC quality-evaluation cost function. The first three weights are proportional to their silicon area in the IC, while w_t reflects our IC speed constraints.

According to the different approaches of multi-objective functions [3] and their efficiency we chose the presented distance function with the variable weight of separate criteria. With this approach it is possible to simplify the conditions or to expose some criteria. As mentioned before, the solution that is closest to the origin of the search space can be found.

2.3 Genetic operators and parameters

In each iteration, e.g., generation, of the algorithm there are four genetic operators that transform the chromosome. They consider data dependencies and the given library of available FUs. Each time after genetic operators transform the chromosome, the chromosome is checked to meet all constraints, considering data dependencies and unit types.

2.3.1 Selection

Upon the cost function values the worse solutions are aborted in the selection step and to ensure equally large population, these solutions are replaced with the best solutions. This ensures best solutions of the given generation to be surely involved in the next generation creation (elitism).

2.3.2 Crossover

In crossover task two approaches are used, each expressing the dominancy of the characteristics. After two crossover points are determined, in the first case the unit information is changed between the two chromosomes and start times are adapted, and in the second case the start times are changed and suitable unit is allocated. So the dominancy is expressed either in FUs or operations start times.

2.3.3 Mutation

Here, we also have two similar approaches to transform the chromosome. In both cases the starting time is changed. Either it is moved to later control steps with the use of faster FUs or it is moved to earlier control steps, if data dependencies allow that, with slower units.

2.3.4 Variation

After two operations are selected and when they are of the same type (e.g., additions), their FUs are switched. If needed also their start times are updated.

3 Test-bench circuits

3.1 Differential equation

Relatively small circuit of differential equation [11] has only 11 operations, but 4 different operation types (6 multiplications, 2 additions, 2 subtractions, 1 comparison), see Figure 1. This circuit is useful when testing libraries with different implementations of the same operation types.

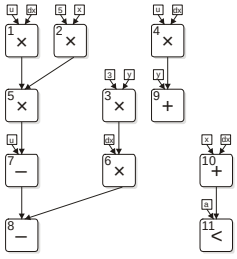

Figure 1: Differential equation

3.2 Elliptic filter

This filter [5] consists of 34 operations, but only two operation types: 26 additions and 8 multiplications (Figure 2). The circuit is suitable for comparison due to its size and operation dependencies, since they form two independent similar critical paths both influencing the circuit delay.

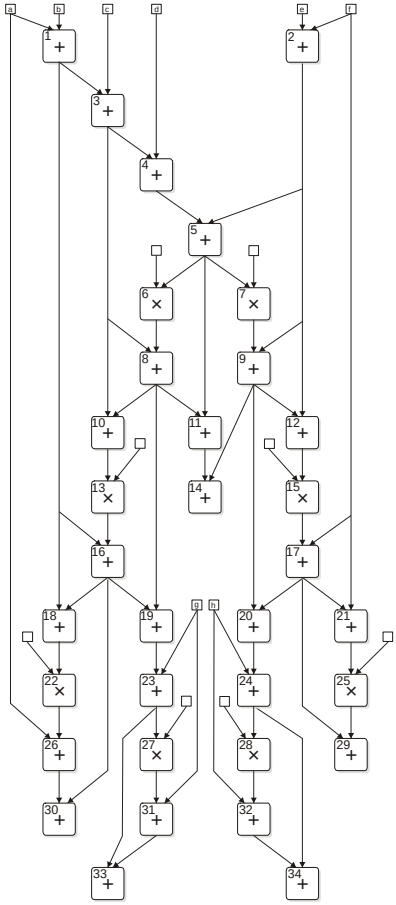

Figure 2: Fifth-order elliptic filter

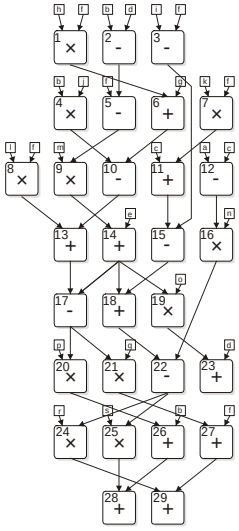

Figure 3: Bandpass filter

3.3 Bandpass filter

One of the implementations of the bandpass filter [5] is the circuit used for our evaluation. It consists of 29 operations; 11 multiplications, 10 additions and 8 subtractions (Figure 3). Due to data dependencies almost all operations influence the circuit delay.

3.4 Least mean square filter

This filter for signal adaptation (noise reduction) is based upon least mean square method [2]. It consists of 47 operations; 24 multiplications and 23 additions (Figure 4). This test-bench circuit is useful due to its size and unique data dependencies.

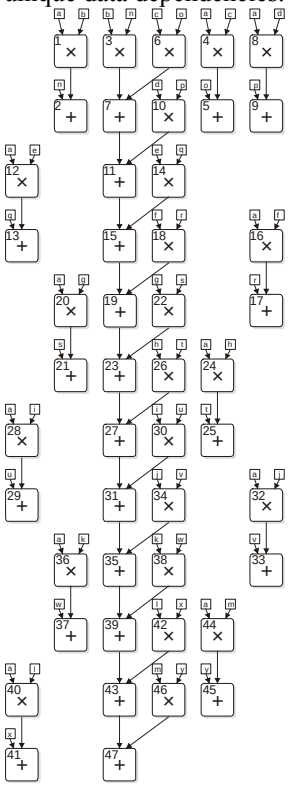

Figure 4: Least mean square filter

4 Evaluation

Considering 18750 different schedules of each circuit and different combinations of parameters, we statistically compared the results according to their cost function (Eq. 1). For each of described four test-bench circuits we made a set of 3125 different combinations of parameters (generations, populations, crossover, mutation and variation). We repeated the optimization process with each combination five times to reduce the influence of statistical error and to get the average fitness of solutions obtained by each combination of parameters.

The solutions with fitnesses of top 20% of all fitnesses for a certain circuit were defined as high quality solutions and solutions with bottom 20% of fitnesses were defined as low quality solutions

To ensure most solutions being time-constrained (executed in shortest possible time) the weight w_t was set to extremely high value.

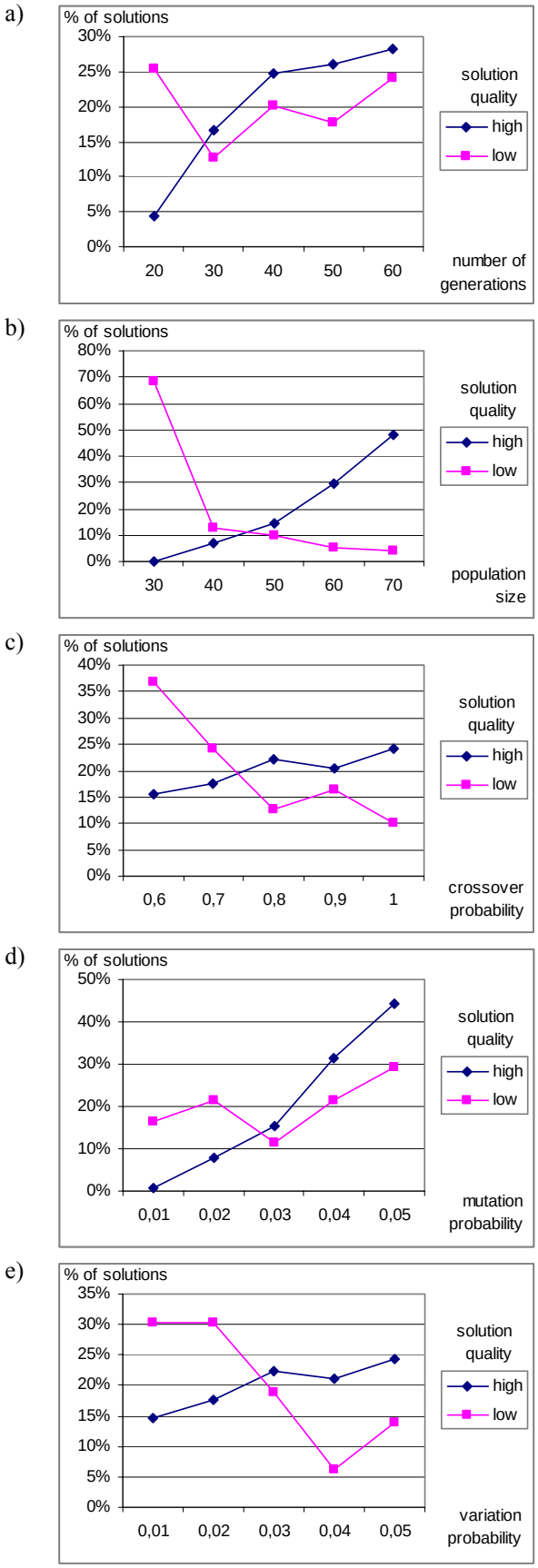

Figure 5: Differential equation

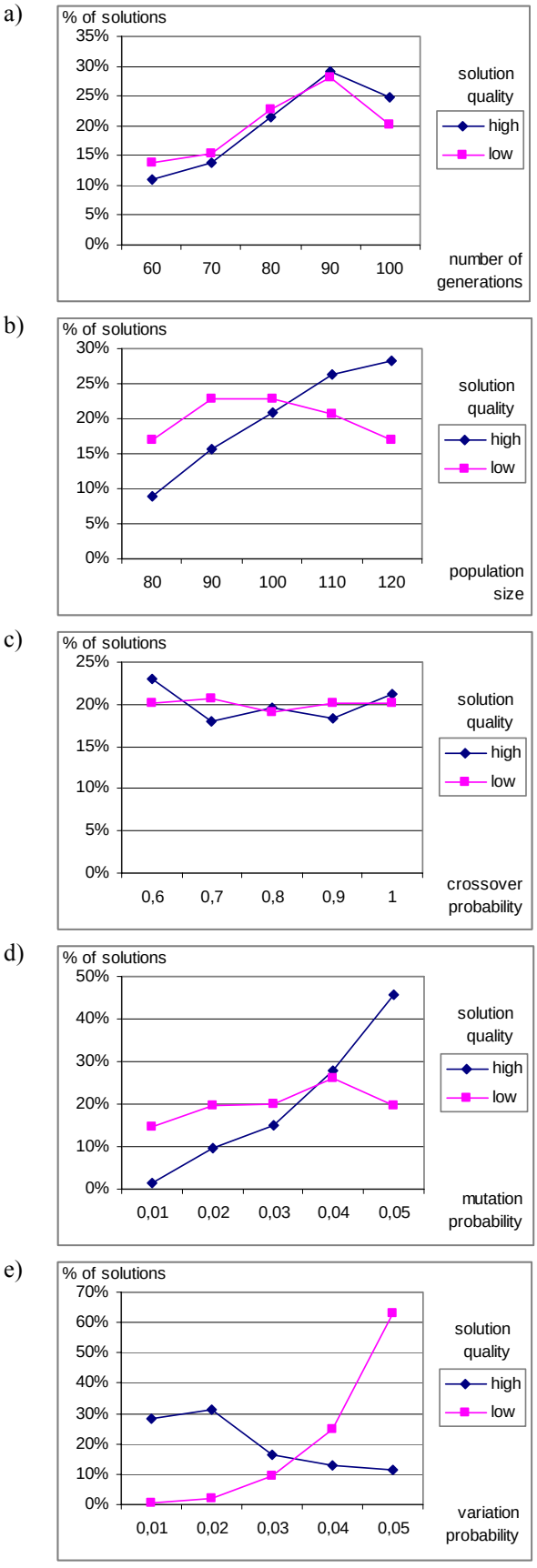

Figure 6: Fifth-order elliptic filter

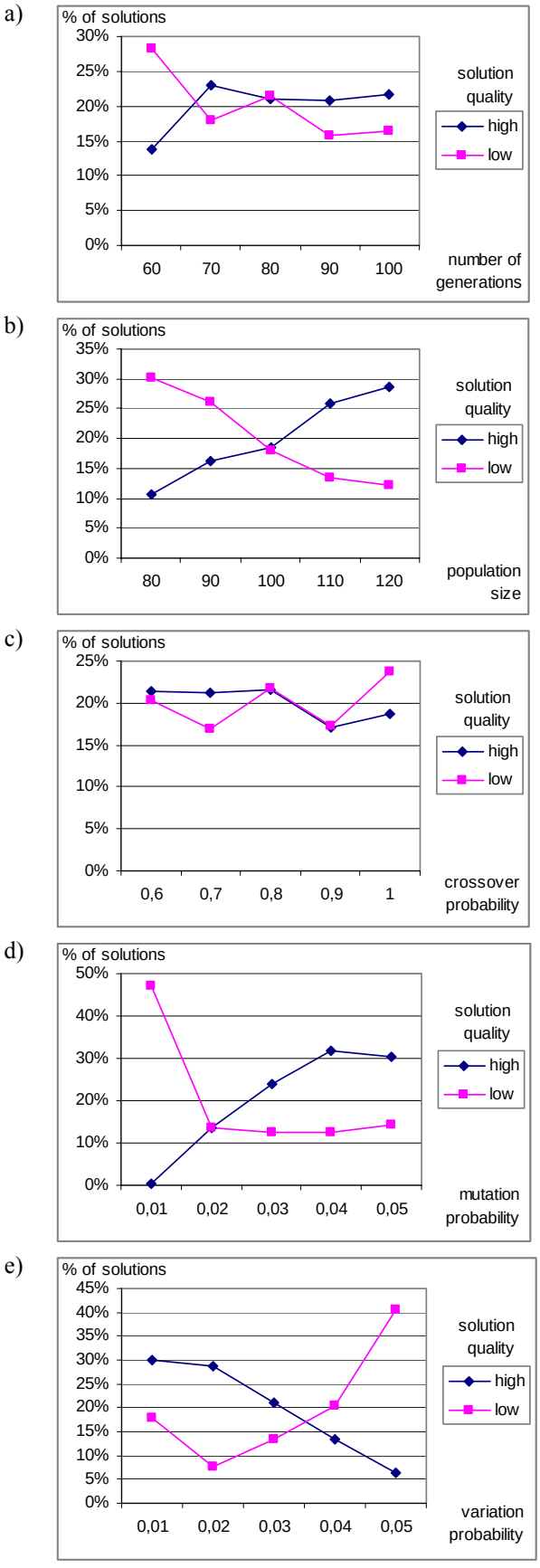

Figure 7: Bandpass filter

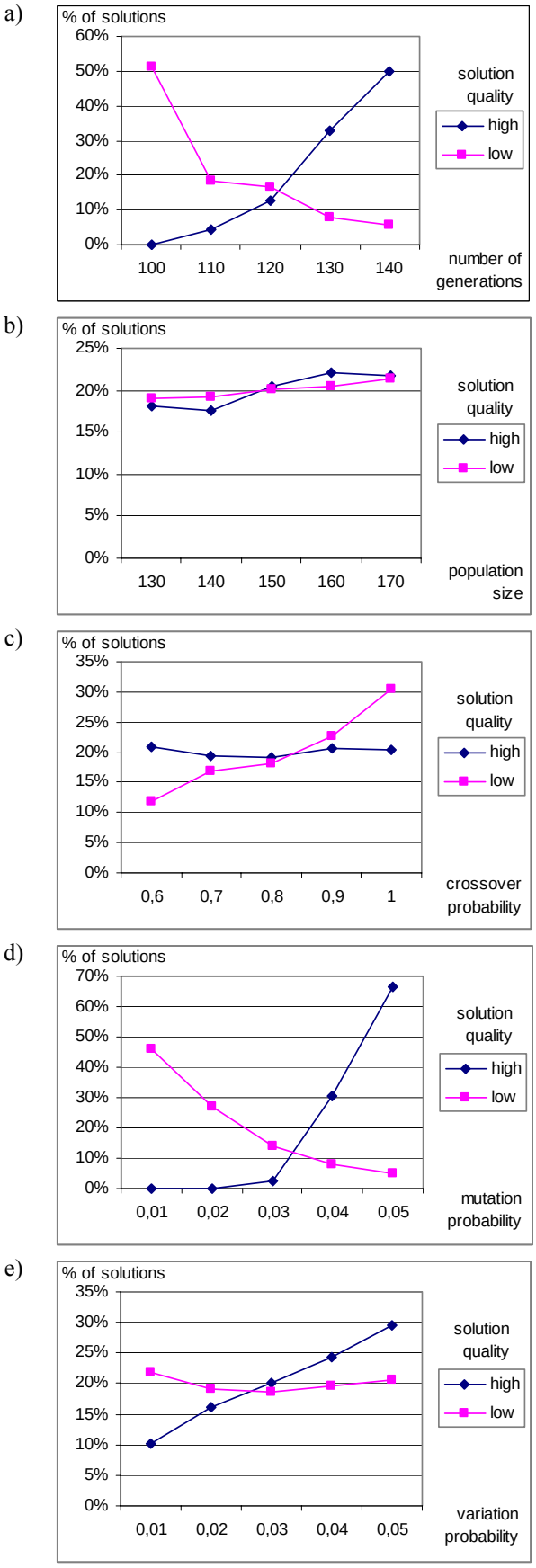

Figure 8: Least mean square filter

As presented in Figures 5, 6, 7, and 8, solutions with high quality are mostly obtained by the following values of parameters: probability of crossover is 0.7, probability of mutation is 0.04, and probability of variation is 0.03. Besides, considering the circuits sizes the number of generations and population size should be set to 3 times and 4 times of a circuit size, respectively.

The values of parameters in this combination are named as optimal values. These optimal values are determined upon the percentage of solutions with certain parameters among high quality solutions. The parameter value, to be considered as optimal, should have at least 25% share among high quality solutions, while it should have less than 10% share among low quality solutions. Of course, there are some minor deviations but in general we can define some average values of genetic operator's parameters when working with high-level IC design.

5 Conclusion

As presented there is a lot of work to fine-tune the proper values of the genetic operators. To achieve compatible results in optimization of the used circuits it is appropriate to use the values obtained by our investigation.

Generally, the quality of solution is always influenced by parameters and the problem itself. Therefore, it is important to perform this kind of evaluation each time we are in search of the optimal values of the genetic operators for some new problem to be solved.

References

[1] T. Bäck, Evolutionary Algorithms in Theory and Practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford University Press, 1996.

[2] J. Benesty, P. Duhamel, A Fast Exact Least Suare Adaptive Algorithm, IEEE Transactions on Signal Processing 40, 1992, pp. 2904-2920.

[3] C. A. Coello Coello, A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques, Knowledge and Information Systems 1, 1999, pp. 269-308.

[4] B. Filipič, J. Štrancar, Tuning EPR spectral parameters with a genetic algorithm. Applied soft computing 1, 2001, pp. 83-90.

[5] G. W. Grewal, T. C. Wilson, An Enhanced Genetic Algorithm for Solving the High-Level Synthesis Problems of Scheduling, Allocation, and Binding, Intl. Journal of Computational Intelligence and Applications, 1, 2001, pp. 91-110.

[6] S. Kirkpatrick, C. D. Gelatt, M. P. Vecchi, Optimization by simulated annealing, Science 220, 1983, pp. 671–680.

[7] F. G. Lobo, The parameter-less genetic algorithm: Rational and automated parameter selection for simplified genetic algorithm operation, Ph.D. thesis, University of Lisbon, Portugal, 2000.

[8] G. Papa, Concurrent operation scheduling and unit allocation with an evolutionary technique in the process of integrated-circuit design, Ph.D. thesis, Faculty of Electrical Engineering, University of Ljubljana, Slovenia, 2002.

[9] G. Papa, B. Koroušić-Seljak, B. Benedičič, T. Kmecl, Universal Motor Efficiency Improvement using Evolutionary Optimization, IEEE Transactions on Industrial Electronics 50, 2003, pp. 602-611.

[10] G. Papa, J. Šilc, Automatic Large-Scale Integrated Circuit Synthesis Using Allocation-Based Scheduling Algorithm, Microprocessors and Microsystems 26, 2002, pp. 139-147.

[11] P. G. Paulin, J. P. Knight, E. F. Girczyc, HAL: A Multiparadigm Approach to Automatic Data Path Synthesis, Proc. 23rd ACM/IEEE Design Automation Conference, Las Vegas, USA, June 1986, pp. 263-270.